Week 2- investigating system requirements

Understanding business case:

1. **Business case**: Justification for a proposal, aligned to organizational missions, objectives and IT needs.
2. Any relation between “Systems projects and business/organization”?
3. Where the organization is at present, where they want to be?
4. A business case should:

– Be comprehensive and easy to understand

– Describe the project clearly, provide the justification to proceed, and estimate the project’s financial impact

1. Questions answered by a business case

* Why are we doing this project?
* How much will it cost and how long will it take?
* Are there any risks involved?
* How will we measure success?
* What alternatives exist?

Systems development request:

Systems request:

* Stronger controls
* More support
* Improved Service
* Better performance
* More information
* Reduced cost

Why investigating requirements?

* Communication efficiency is low

What are requirements?

System Requirements

* **Functional** requirements: the activities the **system must perform**
  + Business uses, functions the users carry out
  + Shown as use cases
* **Non-functional** requirements: other system characteristics
  + Constraints and performance goals

|  |  |  |
| --- | --- | --- |
| Requirement categories | FURPS categories | Example requirements |
| Functional | **F**unctions | Business rules and processes |
| Nonfunctional | **U**sability  **R**eliability  **P**erformance  **S**ecurity | User interface, ease of use  Failure rate, recovery method  Response time, throughput  Assess controls, encryption |

Additional requirement categories

Design constraints

* Specific restrictions for hardware and software

Implementation requirements

* Specific languages, tools, protocols, etc.

Interface requirements

* Interface links to other systems

Physical requirements

* Physical facilities and equipment constraints

Supportability requirements

* Automatic updates and enhancement methods

**Evaluation** of systems requirements

Systems requests are evaluated by a **systems review committee** or a **computer resources committee**

**Systems Request Forms**

◦ Streamline the request process

◦ Ensure consistency

◦ Easy to understand

◦ Include clear instructions

◦ Indicate the required supporting documents

◦ Submitted electronically

Systems Review Committee

* A **broader viewpoint** enables a committee to establish priorities more effectively than an individual

• One person’s **bias** is **less** likely to **affect decisions**

* Disadvantages

• Action on requests must **wait** until the committee meets

• Members might **favor** projects requested by their **own** departments

• **Internal** political differences could **delay** important decisions

Overview of Feasibility

* Feasibility studies can be simple or exhaustive
* Effort required depends on the nature of the request
* Initial fact-finding involves:
  + Studying organizational charts
  + Performing interviews
  + Reviewing current documentation
  + Observing operations
  + Surveying users
* A **feasibility** study examines operational, technical, economic, and schedule factors.
  + **Operational**: will it be easy to learn and use?
  + **Technical**: Do we have the tech resources?
  + **Economic**: will benefits exceed costs?
  + **Schedule**: Can we do it in time?

▪ **Operational Feasibility**

– A proposed system will be used effectively after it has been developed

– Can be affected by organizational culture

– Cannot be accurately measured but requires careful study

– Questions that can help predict a system’s operational feasibility

• Is the project supported by management and users?

• Will the new system result in a workforce reduction?

• Do legal or ethical issues need to be considered?

▪ **Economic Feasibility**

– Projected benefits of a proposed system out-weigh **total cost of ownership (TCO)** – Determination of TCO requires cost analysis of:

• People, including IT staff and users

• Hardware and equipment

• Software

• Formal and informal training

• Licenses and fees

• Consulting expenses

• Facility costs

– **Tangible costs** are measured in dollars

– **Intangible costs** can significantly affect organizational performance

– **Tangible benefits** can result from a decrease in expenses or an increase in revenues

– **Intangible benefits** are important to the company despite the inability to measure them in dollars

▪ **Technical Feasibility**

– Technical resources required to acquire and use the system

– Questions analysts should ask

• Does the company have the necessary hardware, software, and network resources?

• Does the company have the required technical expertise?

• Does the proposed platform have sufficient capacity for future needs?

• Will a prototype be required?

▪ **Schedule Feasibility**

– A project can be implemented in an acceptable time frame

– Issues that can affect schedule feasibility

• Interaction between time and costs

• Can the company or the IT team control the factors that affect schedule feasibility?

• Has management established a firm timetable for the project?

• What conditions must be satisfied during the development of the system?

• Will an accelerated schedule pose any risks?

Evaluating Feasibility

▪ Identify and weed out systems requests that are not feasible

▪ Requests that are not currently feasible can be resubmitted as new hardware, software, or expertise becomes available

Prepare for investigation

▪ **Interaction with Stakeholders**

▪ Stakeholders– persons who have an interest in the successful implementation of the system

- **Internal** Stakeholders– persons within the organization

- **Externa**l stakeholders – persons outside the organization

- **Operational** stakeholders – persons who regularly interact with the system

- **Executive** stakeholders– persons who don’t directly interact, but use the information or have financial interest

▪ Steps in the preliminary investigation

**–** **Understand the problem or opportunity**

• Develop a **business profile** that describes **current business processes and functions**

• Understand how modifications will affect business operations and other information systems

• Identify the **departments, users, and business processes** involved

• Consider using a **fishbone diagram**

**– Define the project scope and constraints**

• Define the specific **boundaries**, or extent, of the project

• Define **project scope** by creating a list with sections called **must do, should do, could do, and won’t do**

• Avoid project creep

– **Project creep**: Process by which projects with very general scope definitions expand gradually, without specific authorization

• Identify constraints

– **Constraint**: A requirement or condition that the system must satisfy or an outcome that the system must achieve

– **Perform fact-finding (重点！)**

there are many techniques:

1. Gather and analyze data

▪ about project usability, costs, benefits, and schedules

▪ Pareto chart (bar diagram), XY chart (scatter diagram)

b) Analyze organization charts

c) Conduct interviews

- Select interviewees

- Design interview questions

- Prepare for the interview

- Conduct the interview

- Post interview follow-up

d) Review documentation

e) Observe operations

f) Conduct a user survey

g) **Joint Application Development**

Key stakeholders – facilitator, scribe, participants.

**– Analyse project usability, cost, benefit, and schedule data**

• Factors to consider

– What information must be obtained, and how will it be gathered and analyzed?

– Who will conduct the interviews? How many people will be interviewed?

– Will a survey be conducted? Who will be involved? How much time will it take to tabulate the results?

– How much will it cost to analyze the information and prepare a report with findings and recommendations?

**– Evaluate feasibility**

• Operational feasibility

• Technical feasibility

• Economic feasibility

• Schedule feasibility

**– Present results and recommendations to management**

• Prepare a report that includes:

– An evaluation of the systems request

– An estimate of costs and benefits

– A case for action

• Format of a report

– Introduction

– Systems request summary

– Findings

– Recommendations

– Project roles

– Time and costs estimates

– Expected benefits

– Appendix

System Requirements Prioritization

▪ Requirements must be **SMART**

- **S**pecific

- **M**easurable

- **A**ttainable

- **R**ealistic

- **T**ime-bound

▪ Techniques— Must, Should, Could, Won’t or Would

▪ Factors that Affect Priority

– Will the proposed system **reduce costs**?

– Will the system **increase revenue** for the company?

– Will the systems project result in more information or produce **better results**?

– Will the system **serve customers better**?

– Will the system **serve the organization better**?

– Can the project be implemented in a **reasonable time period**?

– Are the necessary financial, human, and technical **resources available**?

▪ Discretionary and Nondiscretionary Projects

– **Discretionary projects**: Projects where **management has a choice** in implementing them

– **Nondiscretionary projects**: **Management has no choice** in implementing a project

• Most of these projects are predictable

– Annual updates to payroll

– Tax percentages

– Quarterly changes

Week6 Data and Process Modeling（重点知识）

Learning Objectives

▪ **Describe** **data and process modeling concepts and tools**, including **data flow diagrams**, a **data dictionary**, and **process descriptions**

▪ **Describe** the **symbols** used in data flow diagrams and **explain** the **rules** for their use

▪ **Draw data flow diagrams** in a sequence, from general to specific

▪ Explain how to **level and balance** a set of data flow diagrams

▪ Describe how a **data dictionary** is used and what it **contains**

▪ Use process description tools, including **structured English**, **decision tables**, and **decision trees**

▪ Describe the relationship between **logical and physical** models

Overview of Data and Process Modeling Tools

▪ **Data flow diagram** (DFD) - Uses various symbols to show how the system transforms input data into useful information

**Data Flow Diagrams**

▪ A data flow diagram (DFD) **shows how data moves through an information system** but does **not show program logic or processing steps**

▪ A set of DFDs provides a logical model that **shows what the system does**, **not how it does**

▪ DFD **Symbols**

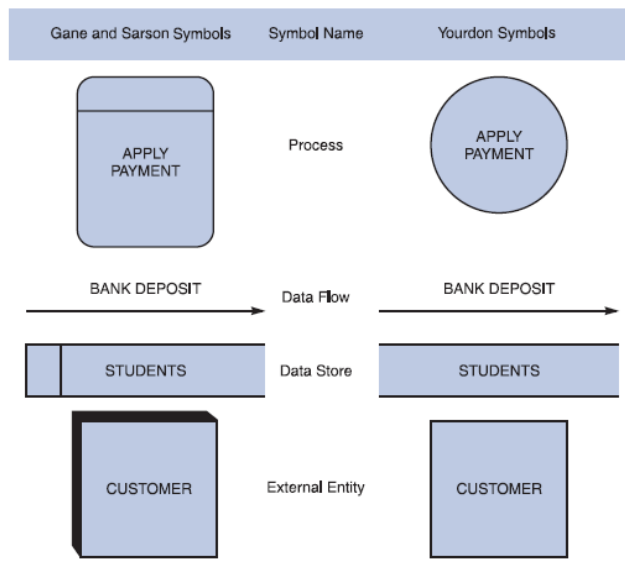
– Four basic symbols represent processes, data flows, data stores, and entities

– **Gane** and **Sarson**: Used in data flow diagrams

• Processes, data flows, data stores, and external entities all have a unique symbol

– **Yourdon**: Used in data flow diagrams

• Processes, data flows, data stores, and external entities each have a unique symbol

例子：

**Process** Symbol （圈 代表信息处理与转化）

• Must have at least one input and at least one output

• Contains **business logic** that transforms the data

• Process name identifies its function (verb)

• Examples: “apply rent payment” or “calculate commission”

• In DFDs, a process symbol can be referred to as a **black box**

**Data Flow** Symbol（箭头 代表某个数据的流动方向）

◦ Represents one or more data items

◦ The symbol for a data flow is a line with a single or double arrowhead

– Following data flow and process combinations must be avoided

◦ Spontaneous generation （两边往外 没有输入）

◦ Black holes （两边往里 没有输出）

◦ Gray holes （输入和输出没关系）

Data Store symbol （长条 代表数据库）

• Represent data that the system stores

• A DFD does not show the detailed contents of a data store — the specific structure and data elements are defined in the data dictionary

• A data store must be connected to a process with a data flow

• 必须至少一进一出， 必须链接“圆”

Entity Symbol（方块 代表与系统互动的外界元素）

• Shows how the system interfaces with the outside world

• A DFD shows only external entities that provide data to the system or receive output from the system

• DFD entities also are called terminators because they are data origins or final destinations

• Each entity must be connected to a process by a data flow（必须通过箭头链接圆 不能链接长条或自己）

**Creating a Set of DFDs（如何画dfd）**

▪ Guidelines for Drawing DFDs

– Draw the context diagram so that it fits on one page

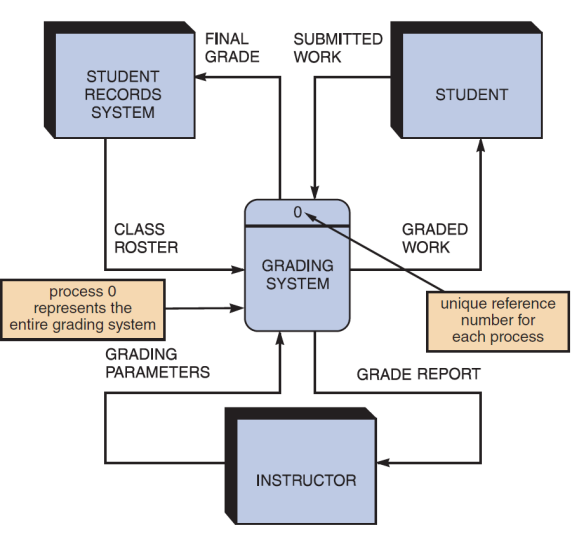
– Use the name of the information system as the process name in the context diagram

– Use unique names within each set of symbols

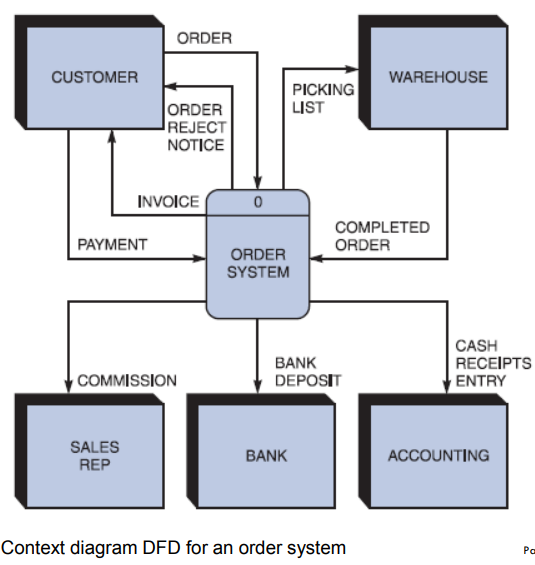
– Do not cross lines

– Provide a unique name and reference number for each process

– Ensure that the model is accurate, easy to understand, and meets the needs of its users

例子：

**▪ Step 1: Draw a Context Diagram**



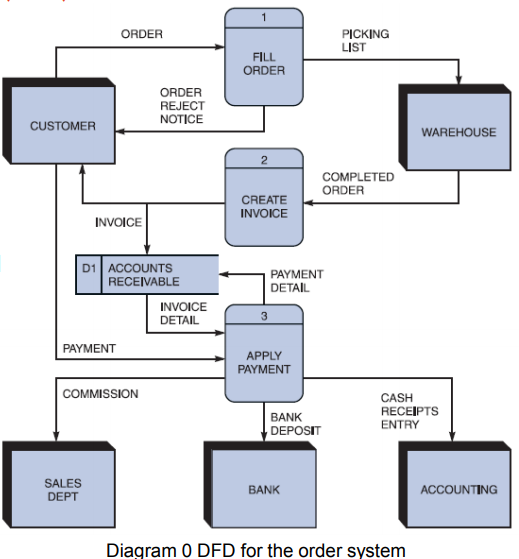
▪ **Step 2: Draw a Diagram 0 DFD**

– If same data flows in both directions, you can use a double-headed arrow

– Diagram 0 is an exploded view of process 0

– Parent diagram

– Child diagram

– Functional primitive

▪ **Step 3: Draw the Lower Level Diagrams（1.1 1.2 1.3构成的小图 解构一个大圆）**

▪ Must use **leveling**（进步划分 把一个圆解构成一张图） and **balancing（让小图的输入和输出和大图中的圆对应的一致）** techniques

▪ **Leveling** examples

– Uses a series of increasingly detailed DFDs to describe an information system

– Exploding, partitioning, or decomposing

**Data Dictionary**

▪ A data dictionary, or data repository, is a central **storehouse of** information about a **system’s data**

▪ An analyst uses the data dictionary to **collect, document, and organize specific facts** about a system

▪ Defines and **describes all** **data elements** and meaningful **combinations of data elements**

**What it contains:**

▪ **Data element**: Smallest piece of data that has meaning within an information system

– Called **data item** or **field**

– Are combined into records, also called data structures

▪ **Record**: Meaningful combination of related data elements that is included in a data flow or retained in a data store

– Called **data structures**

▪ Using CASE Tools for Documentation

– More complex the system, more difficult it is to maintain full and accurate documentation

– Modern CASE tools simplify the task

– A CASE repository ensures data consistency

▪ Documenting the **Data Elements** （记住每个东西都含什么不含什么 考试会出选择）

– Every data element in the data dictionary should be documented

– **Objective**: To provide clear, comprehensive information about the data and processes that make up a system

▪ Contains:

– Data element name and label

– **Alias**

– Type and length

– Default value

– Acceptable values - **Domain** and **validity rules**

– Source

– Security

– Responsible user(s)

– Description and comments

▪ Documenting the **Data Stores**

◦ Data store name or label

◦ Description

◦ Alternate name(s)

◦ Attributes

◦ Volume and frequency

▪ Documenting the **Entities** - Data dictionary describes all external entities that interact with the system

– Characteristics include

• Entity name

• Description

• Alternate name(s)

• Input data flows

• Output data flows

▪ Documenting the **Records**

– Record or data structure name

– Definition or description

– Alternate name(s)

– Attributes

▪ **Data Dictionary Reports** :Following can be obtained

– Alphabetized list of all data elements by name

– Report describing each data element and indicating the user or department that is responsible for data entry, updating, or deletion

– Report of all data flows and data stores that use a particular data element

– Detailed reports showing all characteristics of data elements, records, data flows, processes, or any other selected item stored in the data

**Process Description Tools**

▪ **Process description**: Documents the details of a functional primitive (at the lowest level of the DFDs and can be decomposed no further) and represents a specific set of processing steps and business logic

▪ Tools - structured English, decision tables, and decision trees

▪ Used in object-oriented development

– O-O analysis - combines data and the processes that act on the data into things called objects, and similar objects can be grouped together into classes

– O-O processes are called methods

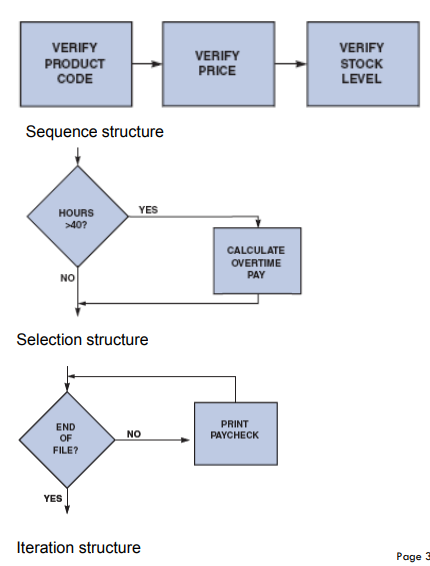
▪ **Modular Design**

– Based on combinations of three **logical structures**, sometimes called **control structures**, which serve as building blocks for the process

• Sequence

• Selection

• Iteration – looping



▪ **Structured English**

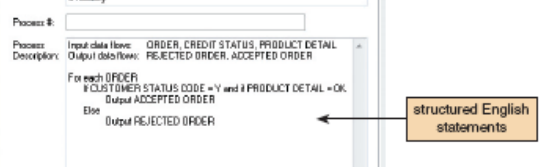
– Rules

• Use only the **three building blocks of sequence, selection, and iteration**

• Use indentation for readability

• Use a limited vocabulary

– standard terms used in the data dictionary

– Specific words that describe the processing rules

▪ **Decision Tables**

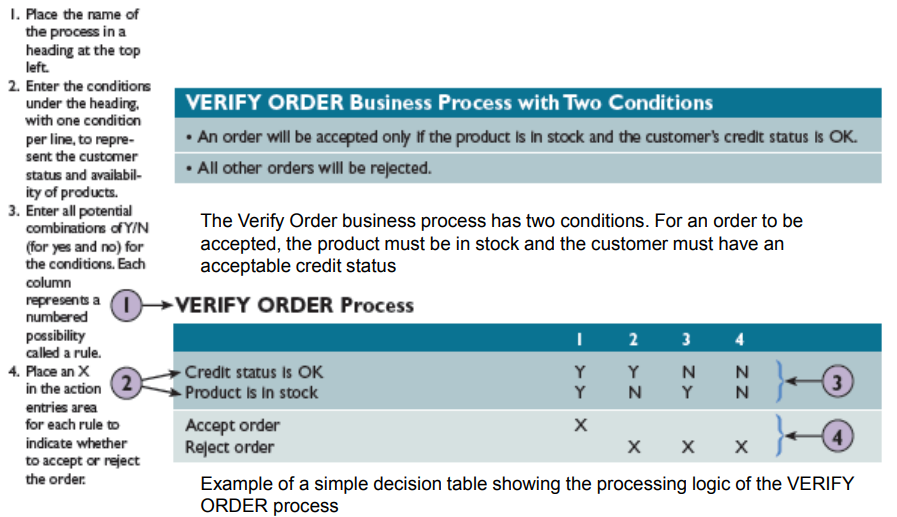
– Show a logical structure, with all possible combinations of conditions and resulting actions

• Every possible outcome should be considered to ensure that nothing has been overlooked

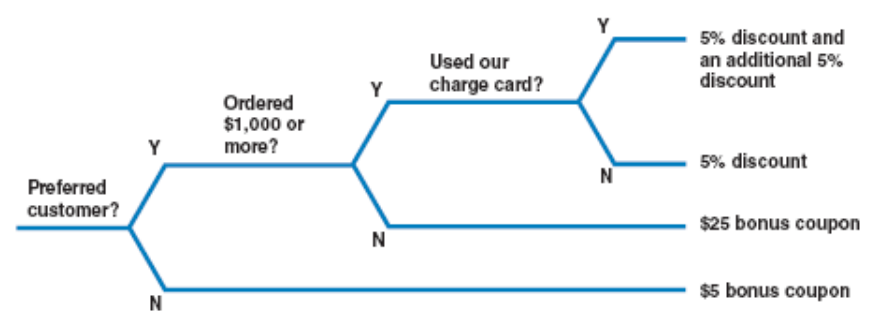
– Number of rules doubles each time a condition is added

– Can have more than two possible outcomes

– Are the best way to describe a complex set of conditions



▪ **Decision Trees**



**Logical and Physical Models**

▪ **Logical Model**: Shows **what the system must do**, regardless of how it will be implemented physically (what)

▪ **Physical Model**: Describes **how the system will be constructed(how)**

▪ While structured analysis tools are used to develop a logical model for a new information system, such tools also can be used to develop physical models of an information system(开发logical的软件也可以开发physical)

▪ Sequence of Models

– Systems analysts **create a physical model of the current system and then develop a logical model of the current system before tackling a logical model of the new system**

**(先how后what)**

• Performing an extra step allows to understand the current system better

▪ **Four-Model Approach**

– Develop:

• A physical model of the current system

• A logical model of the current system

• A logical model of the new system

• A physical model of the new system

– Disadvantage - Additional time and cost

Week7 Object Modeling

Learning Objectives

▪ Explain how **object-oriented analysis** can be used to **describe an information system**

▪ Define **object modeling terms and concepts**, including **objects, attributes, methods, messages, classes, and instances**

▪ **Describe Unified Modeling Language (UML)** tools and techniques including – **use cases, use case diagrams, class diagrams, sequence diagrams, state transition diagrams, and activity diagrams**

Overview of Object-Oriented Analysis

▪ O-O methodology is popular because it **integrates easily with object-oriented programming languages such as Java**

▪ Programmers like **O-O code** because it is **modular, reusable, and easy to maintain**

▪ The end product of O-O analysis is an object model

▪ **Object model**: Represents the information system in terms of **objects and OO concepts**

▪ Object-Oriented Terms and Concepts

– Unified modeling language (UML)

• Method of visualizing and documenting an information system

– **Attributes: Characteristics that describe an object**

– **Methods: Tasks or functions that the object performs**

**– Message: Command to perform a specific function**

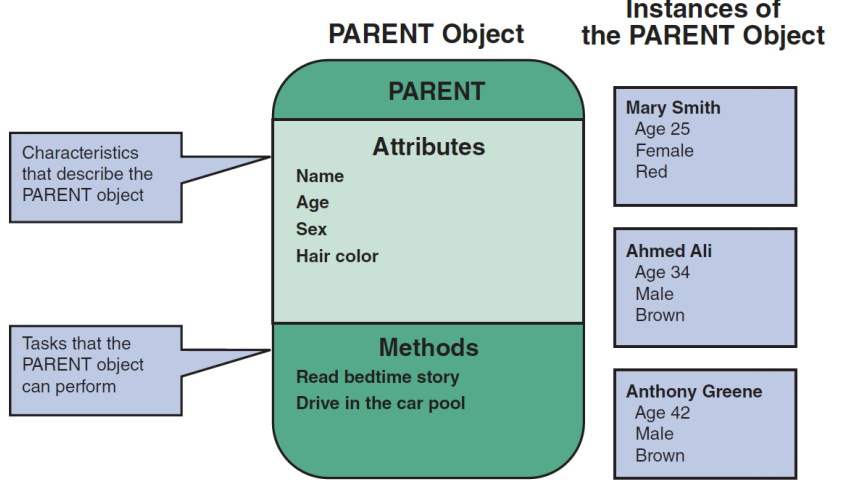
**– A class is a group of similar objects**

**• Instance: Specific member of a class**

▪ **Objects**

– Represented as a rectangle

• The object **name is at the top, followed by the object’s attributes and methods**



▪ **Attributes**

– Describe the characteristics of an object

– The **number** of attributes required depends on:

• **Business requirements** of the information system

• **Requirements of users**

– **Attributes** of an object are defined **during the system development process**

– Objects possess a **state**

• **State**: Describes the **object’s current status**

▪ **Methods**

– **Specific tasks** that an object can perform

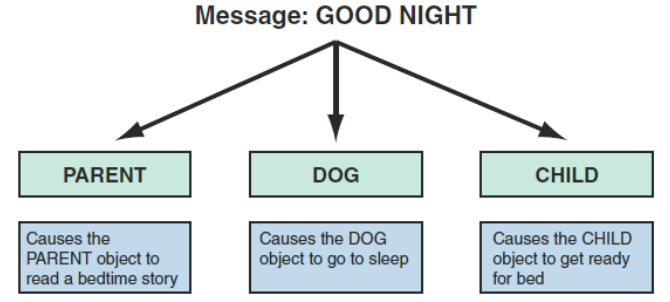
– **Identify** **functions** performed

– **Describe the functions** performed

▪ **Message**

– A **command** that **tells an object to perform a certain method**

– **Polymorphism**: Concept that **a message gives different meanings to different objects**



– A message to the object **triggers changes within the object without specifying how the changes must be carried out**

• An **object** can be viewed as **black box**

– **Encapsulation**: Idea that all **data and methods are self-contained**, as in a black box

▪ **Classes**

– **An object belongs to a group or category called a class**

• All objects within a class share **common attributes and methods**

– **Subclasses**: Categories within a class

– **Super-class**: A class belonging to a general category

Relationships Among Objects and Classes

▪ Relationships

– Enable objects to communicate and interact as they perform business functions and transactions

– Describe what objects need to know about each other

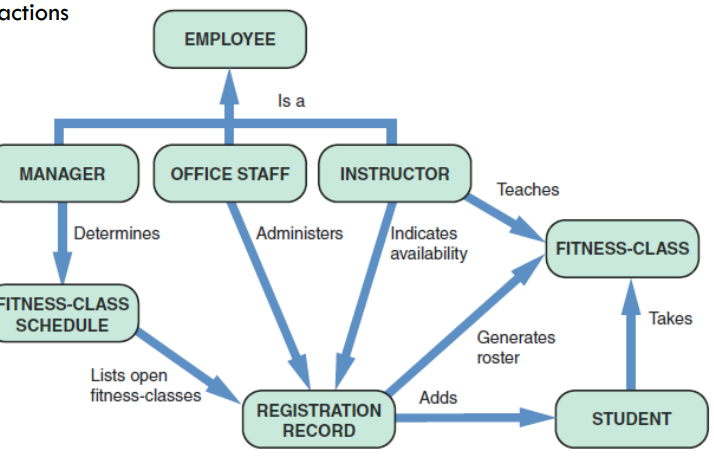
▪ Inheritance （继承）

– The strongest relationship

– Enables an object to derive one or more of its attributes from another object

▪ Object Relationship Diagram

– Displays objects and how they interact to perform business functions and transactions



Object Modeling with the Unified Modeling Language

▪ **Use Case Modeling** (用例模型)

– **Use case**: Represents the **steps in a specific business function or process**

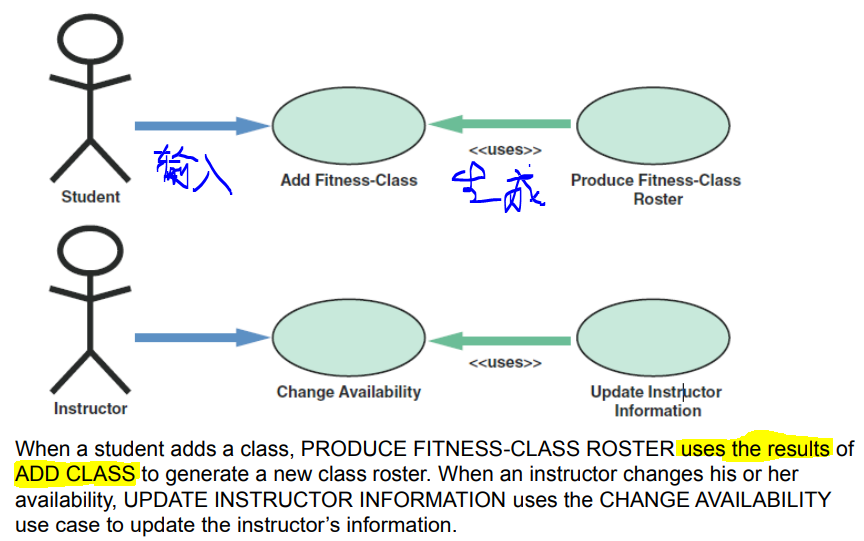
– An **external entity**, called an **actor**, **initiates a use case** by **requesting the system to perform a function** or process

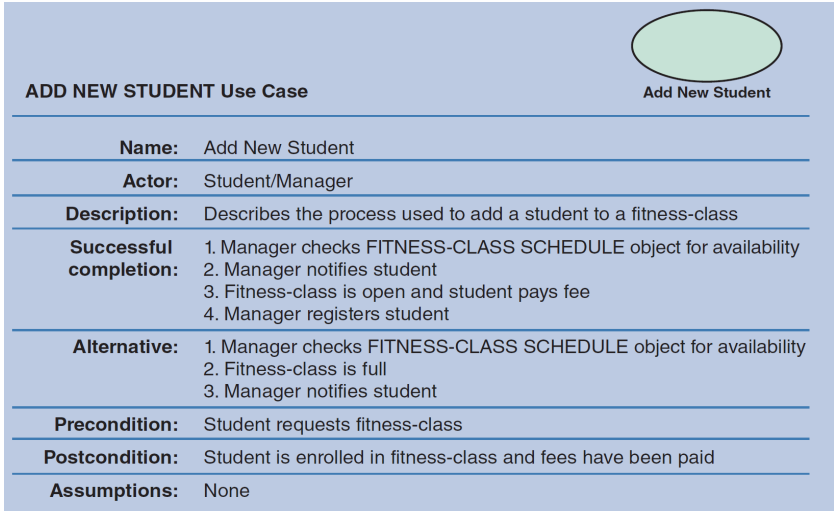
– The **actor** is shown as a **stick figure**, with a **label that identifies the actor’s role**

– **Use case description**: Documents the **name** of the use case, the **actor**, a **description** of the use case

• Provides a **step-by-step** **list** of thetasks and other key descriptions and assumptions

例子：





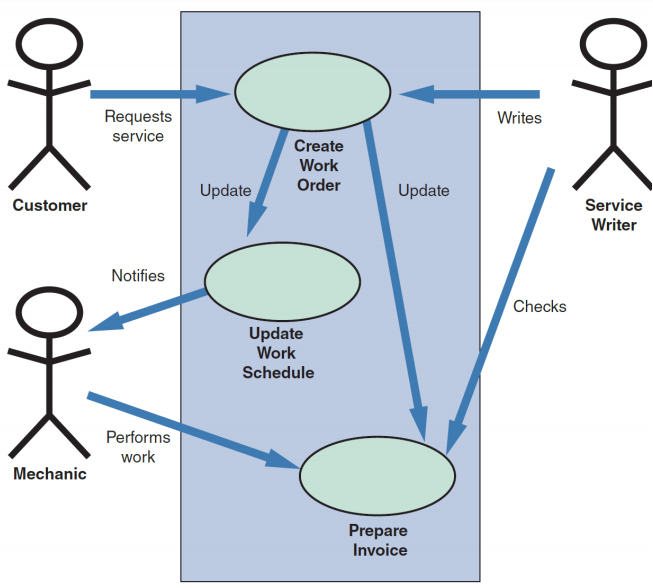
**▪ Use Case Diagrams** （用例图）

– A **visual summary** of several **related use cases** within a system or subsystem

– The first step is to identify the **system boundary** which is represented by a **rectangle**

• **System boundary**: Shows what is **included in the system** (inside the rectangle) and what is **not included in the system** (outside the rectangle)

例子：



▪ **Class Diagrams** （类图）

– Show the object **classes and relationships involved in a use case**

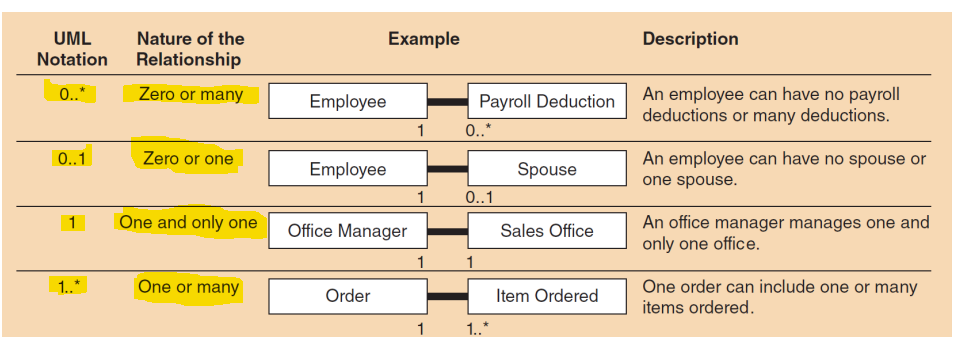
– Each **class** appears as a **rectangle**, with the **class name** at the top, followed by the class’s **attributes and methods**

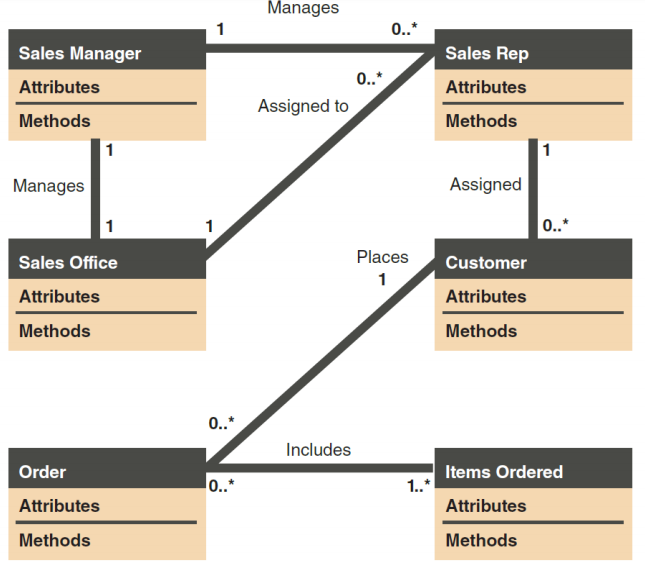
– **Lines** show relationships between classes and have **labels** identifying the action that relates the two classes

– Includes a concept called cardinality

• **Cardinality**: Describes how **instances of one class relate to instances of another class**

**图示：**



**例子：**

（class diagram）

▪ **Sequence Diagrams** （序列图）

– Dynamic **model of a use case**, showing the **interaction among classes** **during a specified time period**

– Graphically document the use case by **showing** the **classes, the messages, and the timing of the messages**

– Include symbols that represent classes, lifelines, messages, and focuses

– Classes

• Send or receive messages

– Shown at the **top** of the sequence diagram

– Lifelines

• Represent the time during which the object above it is able to interact with the other objects in the use case

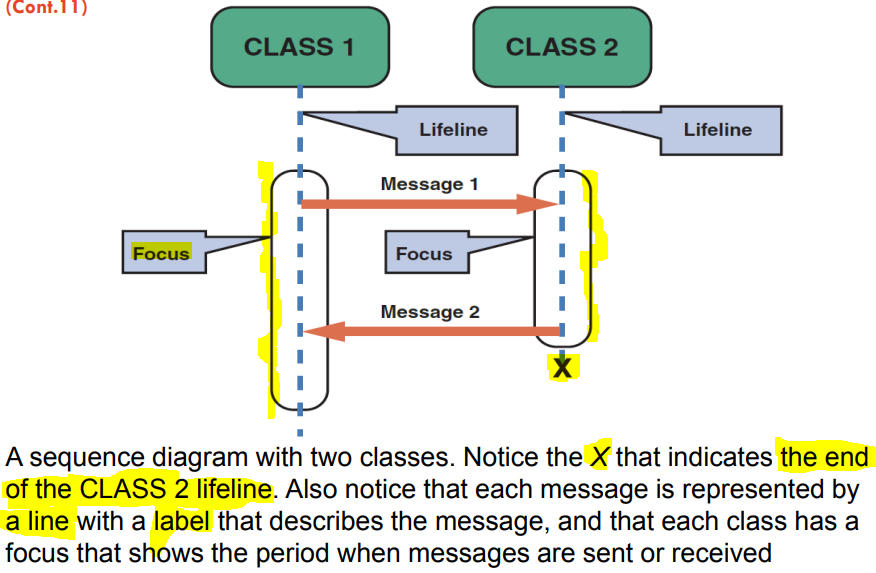
• An **X marks** the end of the lifeline

– Messages

• Include **additional information** about the contents

– Focuses

• Indicate **when** an object sends or receives message

**例子：**

**(sequence diagram)**

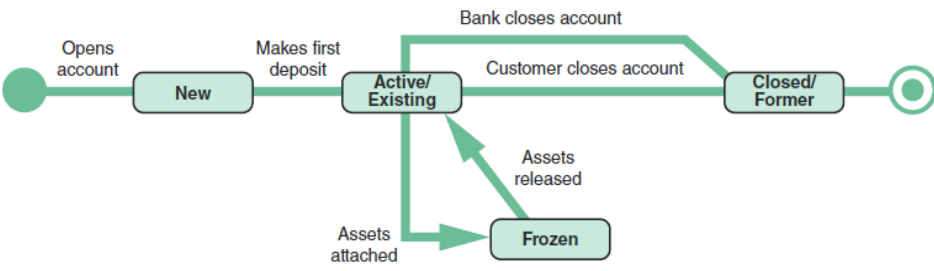
▪ **State Transition Diagrams** （状态转移图）

– Show how an **object changes from one state to another**, depending on **events** that affect the object

– **All possible states** must be documented in the state transition diagram

– States appear as **rounded rectangles** with the state **names inside**

**例子：**



**（状态转换图）**

Week8 User Interface Design

Learning Objectives

▪ Explain the **concept of user interface design** and **humancomputer interaction**

▪ Discuss **Habits of Successful Interface Designers**

▪ Explain **Guidelines for User Interface** Design

▪ Discuss **technology trends**

▪ Discuss **challenges of designing for different devices**

**What is User Interface (UI)?**

▪ The user interface is the **part of the system that you can see, hear and feel**.

▪ Describes **how users interact with a computer system**

▪ **Comprises features that affect two-way communications between the user and the computer**

▪ A physical space that allows human-computer interaction (HCI)

User Interface -- **Usability**

▪ How well the UI is designed affects the usability of the system.

▪ **Usability**: the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

User Interface -- **Interface Types**

▪ **Visual interface**: Graphical User Interface (GUI)

– graphics, colour, typography, etc.

– GUI elements: buttons, icons, prompts, display cards, menus, scroll bars, form elements, etc.

▪ **Auditory interface**: Voice user interface design

▪ **Non-traditional interfaces**: e.g. Motion detection. Gestures. Voice recognition etc.

**Interaction Design**

▪ Designing interactive products to support the way people communicate

– **UI design goes with interaction design** in order to support **usability**.

▪ it can be:

– Human-Computer Interaction (HCI)

– Machine to Machine interaction

– Human-Robot Interaction

– Animal-Computer Interaction

**Designing for users**

▪ **Human-centred design**: Design should support the **user’s needs**.

– **Empathise** with the **user**

– This includes designing for **different** groups of **people**.

▪ **Design for accessibility**: Support usability for users with **reduced abilities**

e.g. short sighted, colour blind, mental and learning disabilities.

▪ **Design for different languages and cultures**: Design for different

– Languages: Scripts, accents, writing direction, etc.

– Culture: design for different social protocols, verbal and non-verbal behaviour, etc.

**Habits of Successful Interface Designers**

} **Understand the Business**

◦ The interface designer must understand:

The underlying business functions

How the system supports individual, departmental, and enterprise goals

} **Maximize Graphical Effectiveness**

◦ A well-designed interface enables rapid learning

} **Think Like a User**

◦ The designer must see the system from a user’s perspective

▪ **Use Models and Prototypes**

– Designers can present initial screen designs to users in the form of a **storyboard**

▪ **Focus on Usability**

– Include main options in the opening screen

– Offer a reasonable number of choices that a user easily can comprehend

▪ **Invite Feedback**

– Monitor system usage and solicit user suggestions

– Determine if system features are being used as intended by observing and surveying users

▪ **Document Everything**

– Document all screen designs for later use by programmers

– User-approved sketches and storyboards can be used to document the user interface

**Guidelines for User Interface Design**

} **Create an Interface That Is Easy to Learn and Use**

◦ Focus on **system design objectives**

◦ Create a design that is **easy to understand and remember**

◦ Provide **commands, actions, and system responses** that are **consistent and predictable**

◦ Allow users to **correct errors easily**

◦ Clearly **label all controls, buttons, and icons**

– Select **familiar images** that users can understand

• Provide on-screen instructions that are logical, concise, and clear

– Show all commands in a list of menu items

• Dim any commands that are not available to the user

– Make it **easy to navigate** or return to any level in the menu structure

▪ **Enhance User Productivity**:make user customise

– If available, consider a **natural language** feature that allows users to type commands or requests in normal text phrases

▪ **Provide Users with Help and Feedback**

– Ensure that help is always available on demand, and information is context-accurate

▪ **Create an Attractive Layout and Design**:highlighting and colour pattern, no complex terms

▪ **Enhance the Interface**

– Opening screen is important as it introduces the application

• The starting point can be a **switchboard** with wellplaced command buttons for navigation

– Use a **command button** to initiate an action

– Try to create customized **menu bars** and toolbars

– Add a shortcut feature that lets a user select a **menu command**

– If variable input data is needed, provide a **dialog box** that explains what is required

– A **toggle button** makes it easy to show on or off status

– Use **list boxes** that display the available choices

– Use an **option button**, or a **radio button**, to control user choices

– If **check boxes** are used to select one or more choices from a group, show the choices with a checkmark or an X

– When dates must be entered, use a **calendar control**

▪ **Focus on Data Entry Screens:**restrict user assess;allow user to change/skip before confirm.

– Use the **form filling** method whenever possible

– Display a sample format like MMDDYY and use an **input mask**

▪ **Use Validation Rules**

– **Sequence check**: Used when the data must be in some predetermined sequence

– **Existence check**: Applies to mandatory data items

– **Data type check**: Tests to ensure that a data item fits the required data type

– **Range check**: Used to verify that data items fall between a specified minimum and maximum value

– **Reasonableness check**: Identifies values that are questionable, but not necessarily wrong

– **Validity check**: Used for data items that must have certain values

– **Combination check**: Performed on two or more fields to ensure that they are consistent or reasonable when considered together

– **Batch controls**: Totals used to verify batch input

▪ **Reduce Input Volume**: only necessary data; no calculated/pre-exist data; no constant data

**Source Document and Form Design**

▪ **Garbage in, garbage out (GIGO)**: Quality of the output depends on the quality of the input

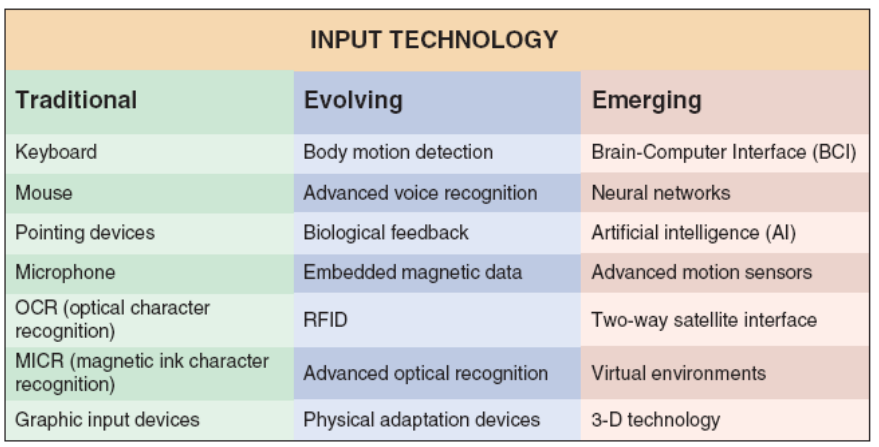
▪ **Source document**: Collects input data, triggers an input action, and provides a record of the original transaction

▪ A good **form layout** makes the form easy to complete and provides enough space

– Information should flow on a form from left to right and top to bottom

**Tech Trends**

Technology Issues: technology is rapidly evolving and with the most disruptive potential.



**Technology Trends: AI**

▪ Dictionary definitions: being human-like rather than becoming human

▪ Build systems that think exactly like humans do (“**strong AI**”)

▪ Just get systems to work without knowing how human reasoning works (“**weak AI**”)

▪ Use **human reasoning** as a **model** but not necessarily the end goal

**Technology Trends: VR**

▪ Immersive experience that provides the opportunity to be digitally transported to a different, place, time or environment.

▪ Comprised of 360 video or rendered content (like video game).

▪ physical objects/controllers that influence the digital experience

**Technology Trends: AR**

▪ Immersive experience that provide the opportunity for digital content, images, 3D models, etc. to render themselves on top / in and **around your real, physical space**.

▪ Is generally comprised of rendered content, but video concepts look promising.

▪ **Content in context** to one’s environment, task, activity.

**Technology Trends: Voice Interaction Technology**

▪ Voice as an interface is fast becoming a major piece of the digital ecosystem.

▪ Takes advantage of advances in **AI**, **speech recognition** and **Natural language processing**

▪ Amazon's Alexa, Google Home, Apple’s Siri, Microsoft’s Cortana etc.

▪ Warnings: **– Alignment of values – Transparency. – Authenticity.**

**Technology trends: Self-Driving Cars**

▪ Autonomous vehicle: A driverless vehicle capable of capable of sensing its environment and navigating without human input.

▪ Objectives: Navigate to a given destination based on passenger-provided instructions

▪ Avoid environmental obstacle

▪ Safely avoid other vehicles

▪ Obey the laws of the road

▪ **Ethical challenges**

**Challenges of designing for different devices**

▪ **Desktop Applications** (operating system specific and strict hardware requirements)

▪ **Web Applications** (client-server model, wide storage and processing resources, interactive, media-rich interfaces)

▪ **Mobile Applications** (limited storage, processing and battery)

▪ Mobile: design challenges

– Slow and Error-Prone Typing

– Less Context

– Inaccurate Clicks

– Poor Connectivity

– Slow Hardware

– Less Storage Capacity

**Tools and Techniques**

▪ **User research**- To find out the needs of the users to create new systems or improve existing systems. Some common techniques include

– ethnography

– user analytics

– usability testing and evaluation (A/B testing)

– interviews, etc.

▪ **Usability heuristics**: Heuristics used to evaluate the UI and interaction design.

▪ **Prototyping**: A model or representation of the system. Can range from low to high level functionality.

– **Low-fidelity prototype** - Mostly includes 2D mock-ups of the system .Low technical skills required. No functionality.

– **Medium-fidelity prototype** - Has the visual look and feel, portrays most of the interactions between the user and the system. No to very little functionality.

– **High-fidelity prototype** - Full or some functionality. Needs a full understanding of technical skills like coding.